Comp 8505-----Final Project

Aman Abdulla

Tao Yuan & Xinghua Wei

A00952007 & A00978597

Introduction

The purpose of this assignment is to design and implement a complete covert application that will allow a user to access a port (that is otherwise closed) on a firewall, communicate with a “disguised” backdoor application. The backdoor application will accept commands and execute them; the results of the command execution will be sent back to the remote client application.

Constraints

There are few constraints I need to follow:

**Part 1**

This portion of the project will deal strictly with the design and implementation of the main backdoor itself.

• This application will accept packets regardless of whether or not any firewall rules are in place once its service port has been opened by a separate application. This means that you will be implementing this part using **libpcap**.

• The application itself will run as a disguised process; you are required to make it as obscure as possible so as to avoid detection.

• The application will only accept those packets that have been authenticated. The authentication will be in the form of an encrypted password in the packet. This can be embedded in the payload or within one of the protocol header fields.

• Once the packet has been authenticated it will extract a command (also encrypted) from the payload portion and execute the command.

• The results of the command will then be sent back to the **attacker** machine (application) using a **covert channel**.

• In addition to sending back requested files, this component will also install a **keylogger** in the compromised system and send the key strokes file to the CNC server.

• One way to configure the application parameters is by means of a configuration file.

**Part 2**

• This portion of the project will implement the file exfiltration and port knocking component for the covert channel to access the **attacker** component and deliver the exfiltrated data.

• The application will watch for the creating of a specific file in a specific directory and when that occurs, it will automatically send the file to the **attacker machine** on the other side.

• The file will be delivered covertly using a special sequence of packets or “knocks”, which the **attacker** machine will authenticate and provide access to the requested port and application.

• Once the exfiltration is complete your application must close access to the ports again.

• Access to the ports may be time-based or controlled by a separate sequence of packets. In other words, the user can remotely specify how and when to close access to the backdoor application.

**Attacker Component**

• The **attacker application** must have all of the features to connect to and control the remote system via the backdoor running on a compromised **victim machine**.

• Aside from simple executing remote commands the overall application must provide an exfiltration function. The user will be able to specify that the remote system search (or watch a directory or file for events) for a particular file and send the contents of that file (including the file containing keystrokes) back to the client application covertly.

• The attacker application will accept and decode the knock sequence, provide access to the port and service that will be accepting all the encrypted data that will contain either the results of the command execution, or the exfiltrated file contents from the server.

• I suggest that all the attacker application parameters be selected and set using a configuration file.

# Design

## Tools

To satisfy all the constraints with Python, the following tools will be used:

* Python 3.8
* Scapy
  + Scapy has a sniff function that is great for getting packets. Sniff has an argument prn that allows users to pass a function that executes with each packet sniffed. Also, unlike traditional TCP\UDP connections, sniff does not need to specify buffer size, which gives me the flexibility to control the packets.
* Setproctitle
  + The setproctitle module allows a process to change its title as displayed by system tools such as ps. Therefore, anyone who is looking at the process table can hardly identify the backdoor process.
* Subprocess
  + Subprocess module allows users to spawn new processes, connect to their input, output and error pipes
  + Popen function takes arguments to set up the new process so the parent can communicate with it via pipes.
  + Stdout and Stderr will be pipes to open and will be where the command executes and where the results store
* Crypto
  + Crypto is a python cryptography toolkit o AES encryption will be used in this assignment
* Watchdog
  + Watchdog is a Python library that can monitor files at the time of its creation or its modification.

## Covert Channel

A covert channel is a mechanism that allows users to send and receive data without the permission of the system. Attackers often make use of this technique to infiltrate the system of their target, either for retrieving data or modifying data. This can be achieved by embedding the data inside unintended fields of the packet, such as the IP ID field, or the IP TTL field.

In our implementation, we have chosen to code our covert channel in Python, and user could choose which protocol he will use.

Diagram

Description automatically generated

The above diagram illustrates how the covert channel work:

1. Attackers open the application and gather user input
2. Attacker machine start construct packets. TTL field in the IP header will be assigned a key that the victim needs to authenticate the key to identify which packets to sniff.
3. Attacker machine then add inn

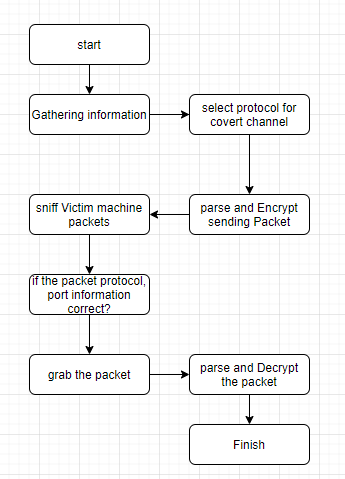
## File Monitoring

## Keylog

Detail Design

Client (Attacker)

For the purpose of explaining how to use the backdoor, I will split the client into three parts: Covert Channel,



Part 1: Covert Channel

The backdoor contains multiple entries that allows the user to enter in IPs, commands, and process names:

• Destination IP: The destination IP is the IP of the target machine where the backdoor server lies. For example, if the server is located on the target machine with an IP of 192.168.1.13, the IP number should be entered here on the client application.

• Source IP: The source IP is the IP of the machine that the packet will contain in the IP headers. If the client user’s IP is 192.168.1.38, entering the IP will allow you to receive back data from the server. The source IP can also be an IP that is not yours. Any oneway commands that do not require data to be sent back, such as creating a directory, or deleting a file, will not require data to be sent back. This will allow the client user to spoof their IP, and if the packet is captured by an unwanted user and analyzed, the source IP shown will be the spoofed IP that the user has entered.

• Process Title: The process title is the title of the backdoor’s name that will be on the target machine. Entering the name “kworker” as the process title will change the process title of the backdoor to “kworker”. This is a powerful feature for when the backdoor is running. It will significantly increase the security and secrecy of your backdoor; if the target machine has all its processes listed, it will display the name that was given by the client user for the backdoor, hiding it behind other kworker processes.

• Information received: there are number of protocols can be selected, UDP, TCP, etc. Therefore, the results of the command will then be sent back to the **attacker** machine (application) using a **covert channel** depends on the protocol.

Once users have all the information, the program will start processing this information and send it to a victim machine.

1. Concatenate the command and process title to one packet

2. Encrypt the packet with AES and UTF-8

3. Send the packet so the victim machine can sniff the packet

4. Sniff the packet in the victim machine using a filter. So, only the information we want would be sniffed.

5. Load the sniffed raw packets and decrypt them into readable information

Part 2: File watching

My backdoor application contains the feature of file watching. The user can enter the directory that they would like to watch for file changes on, providing a relative path to the backdoor server. For example, if the client user wanted to watch for file changes that occurs in the directory “test” which lies in the backdoor directory, the user would simply type in the user interface entry “test”. If the user typed in test/directory, the watch would look for changes that occur in backdoor/test/directory if it exists.

File watcher uses a Python API library called Watchdog that monitors for file system events. I’ve created the class to check for changes occurring such as modification of files, deletion of files, and addition of files.

• Deletion of files: The client user would be notified if the file was deleted or moved to a different directory. In the case of renaming a file, the user would be notified with the name of the file that has been deleted and the name of the new file which was created

• Creation/Addition of files: The client user would be notified if a new file was created or moved in from another folder. The user will be notified with the name of the file that is created.

• Modification of files: In the case that a file has been modified, the user will be notified with the name of the file that has been modified.

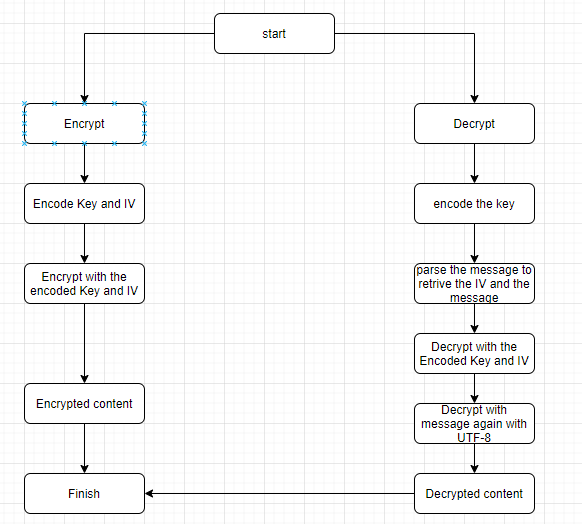
Part 3 : Retrieving Files

The third main feature of the backdoor application is to retrieve files from the target machine. Used in conjunction with the command line feature, the user can enter in the name of a file to retrieve. The user can type in “ls” to list the files that are in the current directory, then type in the file that they want to retrieve. For example, typing in key.log would transfer over the keylogger file. The transfer of files is done via sockets. The user can only select files that reside in the directory of the server application.

Server (victim)

Upon running the server, the keylogger will immediately start recording keystrokes. A file called security.log (for secrecy purposes) that resides in the same directory as the server backdoor will be created and record all keystrokes. The server will begin listening to any commands that are sent by the client and upon receiving commands will process them, sending results back to the client. The server can be shut down from the client by typing “exit” in the command line

Cipher



For cipher, I will use AES with CFB 8 mode (8-bit cipher feedback mode). The following information is required:

Key: fixed data block size of 16 bytes

IV: Initialization Vector is used by several modes to randomize the encryption and produce distinct ciphertexts even if the same plaintext is encrypted multiple times. For CFB mode, it must be 16 bytes long.

Encryption:

1. Encode the key and the initialization vector with UTF-8 because AES in Crypto cannot take a string.

2. Encrypt the message and return the IV and the message

Decryption:

1. Encode the key

2. Parse the message that the former 16 bytes will be the IV and the rest will be the message

3. Decrypt the message with the key and the IV

4. Decode the message again with UTF-8 because it is still in bytes format

Test

|  |  |  |  |
| --- | --- | --- | --- |
| Test# | Description | Results o Test | Pass/Fail |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |